REMARKS

Claims 1-27 are pending in this application. No amendments were made.

Claim Rejections Under 35 USC §102

Applicants respectfully traverse the rejection of claims 1-27 under 35 U.S.C. 102(e) as being anticipated by *Levine et al.*, U.S. Patent No. 5,835,702 ("Levine"). In order for a claim to be anticipated under 35 USC §102, the anticipating reference must disclose at least one embodiment that incorporates all of the claimed elements. See, for example, C.R. Bard, Inc. v. M3 Systems, 48 U.S.P.Q.2d 1225, 1230 (Fed. Cir. 1998) ("When the defense of lack of novelty is based on a printed publication that is asserted to describe the same invention, a finding of anticipation requires that the publication describe all of the elements of the claims, arranged as in the patented device..."); In re Bond, 15 U.S.P.Q2d 1566, 1567 (Fed. Cir. 1990) ("For a prior art reference to anticipate in terms of 35 U.S.C. § 102, every element of the claimed invention must be identically shown in a single reference..."). Thus, in order for the above claims to be anticipated by Levine, Levine must contain every element in claims 1-27.

Claims 1, 9, 17 and 25 are Not Anticipated by Levine:

Applicants respectfully submit that the Office Action has not cited to portions of Levine that teach, disclose or suggest every element of claim 1. For example, the Office Action has not cited to portions of Levine that teach, disclose or suggest at least the elements

> a processor;
>
> an operating system executed by the processor;
>
> a performance counter provider application process executed by the processor;
>
> a performance counter consumer application process executed by the processor;
>
> an application program interface of the operating system, comprising a set of functions including:
>
> a counter registration function called by the performance counter provider application process to allocate a performance counter structure within an address space of the performance counter provider application process, the address space

> designated by the operating system, wherein the counter registration function assigns an access function for retrieving performance counter data from the performance counter structure; and
>
> a counter query function called by the performance counter consumer application process to retrieve counter data from the performance counter structure within the address space of the performance counter provider application process by invoking the access function.

Applicants respectfully submit that the Office Action has not cited to portions of Levine that teach, disclose or suggest every element of claim 9. For example, the Office Action has not cited to portions of Levine that teach, disclose or suggest at least the elements

> providing, via the operating system, an application program interface comprising a set of functions, the set of functions comprising a counter registration API function and a counter query API function;
>
> calling the counter registration API function to allocate a performance counter structure within an address space of the performance counter provider application process, the address space designated by the operating system, wherein the counter registration function assigns an access function for retrieving performance counter data from the performance counter structure; and
>
> calling the counter query API function to retrieve counter data from the performance counter structure within the address space of the counter provider application process by invoking the access function.

Applicants respectfully submit that the Office Action has not cited to portions of Levine that teach, disclose or suggest every element of claim 17. For example, the Office Action has not cited to portions of Levine that teach, disclose or suggest at least the elements

> providing, via the operating system, an application program interface comprising a set of functions, the set of functions comprising a counter registration API function and a counter query API function;
>
> calling the counter registration API function to allocate a performance counter structure within an address space of the performance counter provider application process, the address

> space designated by the operating system, wherein the counter registration function assigns an access function for retrieving performance counter data from the performance counter structure; and
>
> calling the counter query API function to retrieve counter data from the performance counter structure within the address space of the performance counter provider application process by invoking the access function.

Applicants respectfully submit that the Office Action has not cited to portions of Levine that teach, disclose or suggest every element of claim 25. For example, the Office Action has not cited to portions of Levine that teach, disclose or suggest at least the elements

> providing, via the operating system, an application program interface comprising a set of functions;
>
> calling a first function of the set of functions for registering a counter provider associated with a counter provider application process within a repository of counter provider descriptions, wherein each counter provider entry in the repository includes a performance counter structure within an address space of the counter provider application process, the address space designated by the operating system, and an access function for retrieving performance counter data from the performance counter structure;
>
> storing performance counter information within the performance counter structure; and
>
> calling a second function of the set of functions for retrieving the performance counter information from the address space within the counter provider application process via the access function, in response to a performance counter query specifying the counter provider.

Claims 1, 9, 17 and 25 Remarks:

Please note that the following discussion is phrased specifically to pertain to claim 1 for clarity and readability. The amendments and discussion, however, apply equally to claims 9, 17 and 25.

Levine does not anticipate claim 1 as Levine does not disclose all of the elements of claim 1, as the following remarks will illustrate.

1. Third element "a performance counter provider application process executed by the processor": Levine does not anticipate a "performance counter provider application process executed by the processor." As described in the specification, page 2, lines 18-21, a performance counter provider "...generates the raw counter information. The performance counter provider for an application containing performance counters, consists of software written specifically for the particular application with which it is associated." The third element also specifies this counter provider as being an *application process*, which is commonly known as technology that facilitates exchanging messages or data between two or more different software applications, e.g., the virtual interface between two interworking software functions. The application process is executed by the processor.

Levine's performance monitor 50 is shown in Figure 4 as being hardware. Levine's performance monitor 50 "...includes a implementation-dependent number (e.g., 2-8) of counters 51, e.g., PMC1 – PMC8, used to count processor/storage related events. Further included in performance monitor 50 are monitor mode control registers (MMCRn) that establish the function of the counters PMCn..." (Levine, col. 7, lines 35-44). Levine's performance monitor (including its PMCn and MMCRn) is implemented in hardware (Levine, Figure 4) and its interface to software is via interrupts (Levine, Figure 4, references 57 and 260). The interrupts result in performance event counts being tracked in registers PMCn of the performance monitor.

While Levine does describe a performance counter provider, his performance monitor consists of hardware so it cannot be an application process executed by the processor. Levine does not anticipate the first element of claim 1 "a performance counter provider application process executed by the processor."

2. Fourth element "a performance counter consumer application process executed by the processor": Levine does not anticipate "a performance counter consumer application process executed by the processor." As described in the specification, page 2, lines 26-27, a performance counter consumer: "...receives and processes the raw performance counter data provided by the provider and presents it to a user." The second element also specifies this counter consumer as being an *application process*, which is commonly known as technology that facilitates exchanging messages or data between two or

more different software applications, e.g., the virtual interface between two interworking software functions. The application process is executed by the processor.

The problem addressed by Levine is that of how to collect performance data, i.e., by using code points to collect raw counter information into control registers of a performance monitor (Levine, col. 2, lines 50-63). Levine does not address performance counter consuming, i.e., how these counts are made available or *rendered* (as per claim 1's preamble) for counter consumer use. Claim 1, as stated by its preamble, is directed to *rendering* performance counter data, and not collecting it. Thus, Levine and the present application are addressing different problems.

The present application is directed to solving problems with this rendering, such as potential name space conflicts and the requirement authors of performance counter providers to be responsible for counter provision, etc. (specification, page 3, lines 10-16). Levine does not disclose how a performance counter consumer application process, nor how this process may obtain/retrieve collected performance data without name space conflicts or other problems with rendering. Levine merely touches at a broad level regarding actions taken on the performance counter consumer side ("The selected performance monitoring routine is completed and the collected data is analyzed" Levine, col. 9, lines 42-43). However, Levine does not state what entity is retrieving and analyzing the data, or how the retrieval and analysis is performed to mitigate rendering issues. Levine does not provide a solution to these issues as does claim 1 of the present application. He does not positively cite a "performance counter consumer application process executed by the processor."

As Levine does not disclose the second element of claim 1 "performance counter consumer application process", Levine does not anticipate claim 1.

3. Fifth element of claim 1, the application program interface of the operating system: Levine does not disclose the fifth element of claim 1: "an application program interface of the operating system, comprising a set of functions."

Levine's invention does not disclose any application program interfaces supported by an operating system. As commonly known in software, an application program interface is

an interface between two or more different software applications to exchange messages and data. Levine's interface between the performance monitor 50 and the software (Levine, Figure 4) is a hardware-software interface, not an interface between two different software entities. Furthermore, Levine's interface is not "of the operating system." As commonly known, an operating system operates on top of hardware, not the other way around. As the performance monitor of Levine is hardware, it cannot be "of the operating system."

Furthermore, Levine's interface does not consist of a set of functions including a counter registration function and a counter query function. Levine's interface consists of interrupts (Levine Figure 4, references 57 and 260), not a set of functions including a counter registration function and a counter query function.

This makes sense as Levine is not directed to *rendering* or making available performance data for a performance counter consumer; Levine is directed towards *collecting* performance data using interrupts between hardware and software and designated hardware registers (PCMn). On the other hand, a performance counter consumer (as per the above discussion) needs to obtain and process performance data, perhaps for a user, so its execution using application interface programs and functions in software makes sense.

As Levine does not disclose the fifth element of claim 1 "an application program interface supported by an operating system of the system, comprising a set of functions," Levine does not anticipate claim 1.

4. Fifth element of claim 1, the counter registration: Levine does not disclose "a counter registration function called by the performance counter provider application process to allocate a performance counter structure within an address space of the counter provider application process."

As previously discussed, Levine does not disclose the application program interface supported by an operating system comprising a set of functions. The counter registration function is one of the set of functions that make up the application programming interface of the fifth element, so Levine does not disclose the counter registration function.

Furthermore, Levine does not disclose the counter registration element where the element allocates a performance counter structure within an address space of the counter provider application process. The performance counters of Levine are allocated a priori as they consist of hardware: "Counters PMCn and registers MMCRn are typically special purpose registers physically residing on the processor 10" (Levine, col. 7, lines 45-47, and Levine, Figure 4). Levine does not disclose a counter registration function (i.e., an application program interface or software-to-software interface) allocating the PMCn registers. Moreover, the PMCn counters are not allocated within an address space of the counter provider application process; they are allocated within a performance monitor. As Levine does not disclose a counter provider application process, he cannot (and does not) disclose an address space of this counter provider application process to be allocated for performance counters.

Allocating an address space of a counter provider application process for performance counters, as performed by the counter registration function of the present application, provides a tidy solution for the problem of rendering performance counter data. As referenced previously, one of the issues of rendering performance counter data is overlapping name spaces. Performance data may be suspect when various different performance counter providers write to common memory spaces, and counter consumers try to retrieve data from common memory spaces.

As commonly known in the art of software, when an operating system executes an application process, it reserves an address space specifically for that process during the process' lifetime. The novelty of the present application lies in having a counter registration function allocate room in the address space of the counter provider application process. The performance data is thus linked uniquely to the counter provider application itself (as the operating system intrinsically designates address space of the counter provider application to be uniquely owned by that application). Performance data rendering is thus guaranteed to be segregated for that particular counter provider application process, and the data is much more likely to be valid.

In Levine, the MMCRn (re)defines each PCMn for a particular function (Levine, col. 7, lines 42-45). The PCMn registers are not allocated by an application interface program

specifically to the address space of a performance counter provider application process. A physical register must be added for each new process, thus adding extra cost and expense to the system (Levine, col. 7, line 65 – col. 8, line 3). The present application provides benefit over Levine by taking advantage of innate operating system characteristics to reallocate memory for performance counters as application processes live and die, thus not only guarding the validity of data in a unique space for the lifetime of a process, but also allowing efficient reuse of memory space without requiring additional hardware registers.

The Office Action cited Levine at col. 2, lines 51-64 as disclosing this element, but the passage does not disclose a counter registration function being called by a performance counter provider application process. This passage of Levine describes code points being inserted into control register locations. The control register locations are not allocated by a counter registration function application programming interface, as disclosed by claim 1. An API of any sort is not mentioned by Levine, and a performance counter provider application calling an API is not mentioned. Additionally, this section of Levine does not disclose allocating memory for performance data in the operating system-assigned address space of the performance provider application process. The control registers of the performance monitor that collect the data are already pre-defined in hardware.

As Levine does not disclose "a counter registration function called by the performance counter provider application process to allocate a performance counter structure within an address space of the counter provider application process," Levine does not anticipate the counter registration of the fifth element of claim 1.

5. Fifth element of claim 1, the counter query: Levine does not disclose "a counter query function called by the performance counter consumer application process to retrieve counter data from the performance counter structure."

As discussed above, Levine does not disclose the application program interface supported by an operating system comprising a set of functions. The counter query function is one of the set of functions that make up the application programming interface of the fifth element, therefore Levine does not disclose the counter query function.

Furthermore, Levine does not disclose a counter query function being called by a performance counter consumer application process to retrieve collected performance data. As established above, Levine is directed to collecting raw performance data, and does not disclose details for retrieving/rendering this data to a consumer application process. Levine does not even disclose a consumer application process (and has no need to), as Levine is solving a raw performance data collection problem.

The present application, however, is directed to solving the rendering problem and provides a specific solution for data retrieval that complements the counter registration function. By claiming a counter query function supported by the operating system, any performance counter consumer application process may easily have valid performance data rendered to it. The counter query function retrieves the segregated (and therefore, valid) data from the performance data structure that was previously allocated in the address space of the counter provider application process by the counter registration function API. Using an API to retrieve the segregated performance data (to complement the set up by the counter registration function) provides a consistent, complementary abstracted interface that the consumer application process may use to obtain valid performance data for further processing.

The Office Action cited Levine at col. 9, lines 32-55 as disclosing this element, however, the passage does not disclose a counter query function for retrieval of performance counter data that is an API supported by an operating system. This passage does not disclose any APIs at all. Nor does it disclose any mechanism, let alone an API supported by an operating system, to retrieve data from a performance counter structure within an address space of a counter provider application process, as does claim 1. Any performance data retrieved by Levine is fetched from the hardware PCMs of the hardware performance monitor, not an address space of a counter provider application process.

As Levine does not disclose "a counter query function called by the performance counter consumer application process to retrieve counter data from the performance counter structure," Levine does not anticipate the counter query of the fifth element of claim 1.

Claims 1, 9, 17 and 25 Conclusion:

As Levine does not disclose at least the third, fourth and fifth elements of claim 1 (as detailed in Remarks 1, 2 and 3, respectively), and as Levine does not disclose at least the counter registration and counter query of the fifth element of claim 1 (as detailed in Remarks 3 and 4, respectively), Levine does not anticipate each and every element of claim 1. Claim 1 is allowable in light of Levine under 35 USC §102(e).

For at least the same reasons as for claim 1, claims 9, 17 and 25 are also allowable in light of Levine under 35 USC §102(e).

Claims 2-8, 10-16, 18-24 are Not Anticipated by Levine::

Dependent claims 2-8, 10-16 and 18-24 include all elements of their respective independent claims 1, 9 and 17. As Levine does not disclose all the elements of independent claims 1, 9, and 17 discussed above in Remarks 1-4, Levine also does not anticipate dependent claims 2-8, 10-16 and 18-24.

Claim 26 is Not Anticipated by Levine:

Applicants respectfully submit that the Office Action has not cited to portions of Levine that teach, disclose or suggest every element of claim 26. For example, the Office Action has not cited to portions of Levine that teach, disclose or suggest at least the elements

> A performance counter provider application process supported by an operating system of a computer for use in a performance counter system embodying a performance counter provider/consumer model, the performance counter provider application process comprising executable instructions for:
>
> requesting, via an application program interface, the operating system to allocate a memory space within the performance

counter provider application process for a performance counter data structure;

storing performance counter information within the memory space; and

providing access by a registered callback function, invoked by a call to the application program interface, to the memory space containing the performance counter data structure.

1. First element of claim 26: Levine does not disclose the first element of claim 26: a "performance counter provider comprising executable instructions for requesting, via an application program interface, the operating system to allocate a memory space within the performance counter provider application process for a performance counter data structure."

The Office Action cited Levine at col. 6, lines 21-31 as disclosing the first element of claim 26, however, this passage does not disclose a performance counter provider comprising instructions for requesting an operating system via an application program interface. Levine, col. 6, line 21-31 is a general description of how instructions in a processor are sequenced and dispatched to appropriate execution units in the processor. At this point of Levine's detailed description, performance counting has not yet been introduced. No mention of a performance counter provider is positively cited in this passage, nor is an application program interface, i.e., an interface between software entities, described. A request to an operating system is not cited.

Furthermore, no allocation of memory space within the address space of a performance counter provider application process is described. No performance counter provider application process is described, as the entities in the cited passage are not performance counter application processes; they are buffers and sequencers. Therefore, no address space of the performance provider application process is allocated since no performance provider application process has been cited.

2. Third element of claim 26: Levine does not disclose the third element of claim 26: "providing access by a registered callback function, invoked by a call to the application program interface, to the memory space containing the performance counter data structure."

The Office Action cited to Levine at col. 9, lines 17-25 as disclosing this element but his passage does not disclose providing access by a registered callback function invoked by a call to the application program interface. Using careful reading prior to this section to determine the appropriate context, Levine states that "performance monitoring is implemented... through configuration of the performance monitor counters by the monitor mode control registers and performance monitoring data is collected" (Levine, col. 9, lines 12-16). Thus, Levine, col. 9, lines 17-25 is directed to *collecting* performance monitoring data, and not *providing access* to already collected data as in the third element of claim 26. As established earlier, Levine's performance monitor counters (PCMn) and monitor mode control registers (MMCRn) are not application processes or address spaces of application processes. No access to the performance monitoring data is positively cited, let alone using a registered callback function via an API to access the memory space of an application process.

3. Conclusion for claim 26: Levine does not disclose at least the first and third elements of claim 26, as detailed in remarks 1 and 2 above. As discussed in previous remarks, Levine's invention is directed towards collecting raw performance data using a hardware performance monitor and interrupts. In Levine, additional desired performance counts require additional hardware registers to be added (Levine, col. 7, line 65- col. 8, line 3), thus adding extra cost and expense. The present application provides benefit over Levine by taking advantage of innate operating system characteristics to reallocate memory for performance counters as application processes live and die, thus not only guarding the validity of data in a unique space for the lifetime of a process, but also allowing efficient reuse of memory space without requiring additional hardware registers.

As Levine does not disclose each and every element of claim 26, Levine does not anticipate claim 26. Claim 26 is allowable under 35 USC §102(e) in light of Levine.

Claim 27 is Not Anticipated by Levine:

Applicants respectfully submit that the Office Action has not cited to portions of Levine that teach, disclose or suggest every element of claim 27. For example, the Office Action has not cited to portions of Levine that teach, disclose or suggest at least the elements

> a processor,
>
> an operating system executed by the processor;
>
> a performance counter provider application process executed by the processor;
>
> a performance counter consumer application process executed by the processor; and
>
> an operating system performance counter application program interface comprising a first set of functions callable by the performance counter provider application process to register a corresponding performance counter provider in a repository, and a second set of functions for serving requests originating from the performance counter consumer application process to enumerate and access the performance counter provider application process.

For instance, Levine does not disclose the third element of claim 27: "a performance counter provider application process executed by the processor." As discussed in the remarks under claim 1, an application process is executed by the processor and allocated a memory space during its lifetime. Levine does not disclose application processes in his disclosure.

Levine also does not disclose the fourth element of claim 27: "a performance counter consumer application process executed by the processor". As discussed earlier, a performance counter consumer "receives and processes the raw performance counter data provided by the provider and presents it to a user" (specification, page 2, lines 26-27). Levine does not positively cite a performance counter consumer in his specification or claims, nor does he cite a performance counter application process.

Furthermore, Levine does not disclose the fifth element of claim 27: "an operating system performance counter application program interface comprising a first set of functions... and a second set of functions...". As discussed earlier, Levine does not disclose an operating system performance counter application program interface, i.e., an interface between software entities, in this case where one of the entities is an operating system. Levine's performance monitor is implemented in hardware (Levine, Fig. 4) and interfaces with software via interrupts. Furthermore, Levine does not disclose the interface comprising two sets of API functions; his invention is directed to interrupts.

As discussed earlier, Levine's invention is directed towards collecting raw performance data using a hardware performance monitor. Additional desired performance counts require additional hardware registers to be added (Levine, col. 7, line 65- col. 8, line 3) thus adding extra cost and expense. The present application provides benefit over Levine by taking advantage of innate operating system characteristics to reallocate memory for performance counters as application processes live and die, thus not only guarding the validity of data in a unique space for the lifetime of a process, but also allowing efficient reuse of memory space without requiring additional hardware registers.

As Levine does not disclose each and every element of claim 27, Levine does not anticipate claim 27. Claim 27 is allowable under 35 USC §102(e) in light of Levine.

CONCLUSION

In view of the above amendment and arguments, Applicants submit the pending application is in condition for allowance and an early action so indicating is respectfully requested.

The Commissioner is authorized to charge any fee deficiency required by this paper, or credit any overpayment, to Deposit Account No. 13-2855, under Order No. 30835/302623, from which the undersigned is authorized to draw.

Dated: June 19, 2008

Respectfully submitted,

By__/W. J. Kramer #46,229/___
William J. Kramer
Registration No.: 46,229
MARSHALL, GERSTEIN & BORUN LLP
233 S. Wacker Drive, Suite 6300
Sears Tower
Chicago, Illinois 60606-6357
(312) 474-6300
Attorney for Applicant